Contents

2 Programming Report

In each part of the following, I will briefly show the optimization problem I'm solving and the corresponding sketch proof. And I will also present some results that are necessary for analysis.

Before diving into the actual optimization problem, I will first introduce some settings.

Remarks: For code files and some information that are not given in detail in this report, please refer to the attached jupyter notebook.

2.1 Settings

Training Set and Testing Set I randomly choose the training set and testing set with a threshold of 80%, i.e., 80% training set.

One-vs-all Strategy The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. The reason why we use this classification strategy is, each class is only represented by one classifier. And we can know the class by its unique classifier.

2.2 Standard Linear SVM

2.2.1 Optimization Problem

The standard form of linear SVM is given as follows,

$$\begin{aligned}&\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2\\&\text{s.t. } 1-y_i\left(\mathbf{w}^\top\mathbf{x}_i+b\right)\leq 0, \forall i\end{aligned} \tag{1}$$

Since sklearn package does not provide a function with strict separation, we will simulate this using $C = 1e5$. And the optimization problem in sklearn is as follows,

$$\begin{aligned}&\min_{w,b,\zeta} \frac{1}{2}w^Tw + C\sum_{i=1}^{n}\zeta_i\\&\text{s.t. } y_i\left(w^T\phi\left(x_i\right)+b\right)\geq 1-\zeta_i\\&\zeta_i\geq 0, i=1,\ldots,n\end{aligned} \tag{2}$$

C in 2 controls the strength of penalty. We can also regard it as an inverse regularization parameter. A very large C will punish the objective heavily if there's a wrong classification. Hence, we want to reach the optimal case where $\zeta_i = 0$

2.2.2 Results and Analysis

Errors of Linear SVM We define errors as

$$1 - \frac{\text{\# correct predictions}}{\text{\# total predictions}}$$

. Hence we have the following result.

- Training error: 0.0
- Testing error: 0.05555555555555558

Coefficient of each Feature We report the coefficient of each feature in Figure 3

	class_0	class_1	class_2
features			
alcohol	1.124830	-1.225506	0.320996
malic_acid	0.276942	-0.627616	0.287007
ash	2.185495	-2.339713	0.059123
alcalinity_of_ash	-0.291461	0.315948	-0.075431
magnesium	0.008947	-0.039717	0.044751
total_phenols	0.338546	0.164319	-0.197533
flavanoids	0.722289	0.355219	-1.500230
nonflavanoid_phenols	0.538344	0.315653	-0.149030
proanthocyanins	-0.523398	1.036353	-0.543082
color_intensity	-0.157918	-1.583835	0.920324
hue	-0.350123	1.165371	-0.509807
od280/od315_of_diluted_wines	0.665703	-0.205745	-0.712546
proline	0.003486	-0.009693	-0.000774
intercept	-21.239481	30.403958	-7.015709

Figure 3: Coefficient and intercept of linear SVM model

Indices of Support Vectors

- Indices of support vectors in Class 0: [6 78 85 112 130 136 3 44 124]
- Indices of support vectors in Class 1: [8 13 44 104 132 17 19 40 53 78 112 119]
- Indices of support vectors in Class 2: [40 83 93 100 121 10 48 73 79 104 125 132]

2.3 SVM with Slack Variables

2.3.1 Optimization Problem

Based on 1, we add slack variables in the objective function. Now the optimization problem becomes,

$$\begin{aligned}&\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i}^{m} \xi_i \\ &\text{s.t. } 1 - \xi_i - y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \le 0, \quad -\xi_i \le 0, \quad \forall i\end{aligned} \tag{3}$$

In our code, we first separate the ys into 3 binary classes, so that we can compute the values of slack variables based on that. Then, we compute the values of slack variables by

$$\xi_i = 1 - y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right)$$

And we dropped the negative values, and further the variable with all negative values in one input pair.

And in this scenario, we want to test the effect of the strength of punishment C. Below are the results.

2.3.2 Results and Analysis

Errors of SVM with Slack Variables Previously, we defined how to measure the error. In this part, I will report the errors for all possible C I tested. See Table 1. We

Table 1: Errors of SVM with slack variables

C	train_error	test_error
0.1	0.021127	0.055556
0.2	0.014085	0.027778
0.3	0.014085	0.027778
0.4	0.014085	0.027778
0.5	0.007042	0.027778
0.6	0.007042	0.027778
0.7	0.007042	0.027778
0.8	0.007042	0.027778
0.9	0.007042	0.027778
1.0	0.0	0.027778

visualize the above results (see Figure 4 and 5). We know that, if we increase the strength of the penalty, the error will decrease heavily at the very beginning stage. And when the strength is high enough, increasing the penalty may not affect the errors.

Coefficient of each Feature See Figure 6[1] for the coefficient report.

[1] If you cannot see this figure clearly, please refer to the jupyter notebook.

Figure 4: Train error

Figure 5: Test error

	index	alcohol	malic_a...	ash	alcalinit...	magnes...	total_p...	flavanoi...	nonflav...	proanth...	color_in...	hue	od280/...	proline	intercept
0	0.1,class_0	0.3231742...	0.2252242...	0.1784318...	-0.189104...	-0.004850...	0.0679097...	0.3165254...	0.0156674...	-0.013364...	0.0782221...	0.0011694...	0.3918227...	0.0064181...	-8.763968...
1	0.1,class_1	-0.471803...	-0.439733...	-0.170343...	0.0560471...	-0.000833...	0.0761813...	0.320055...	-0.001875...	0.2694152...	-0.700451...	0.2097167...	0.0513486...	-0.004503...	10.853989...
2	0.1,class_2	0.1663260...	0.2871858...	0.0367717...	0.0451078...	0.0035674...	-0.096886...	-0.624539...	-0.003998...	-0.272308...	0.4373024...	-0.176402...	-0.366500...	-0.000874...	-3.476453...
3	0.2,class_0	0.5110352...	0.426875...	0.3522092...	-0.200886...	-0.006908...	0.0385081...	0.496803...	0.0327233...	-0.071468...	-0.060589...	-0.004139...	0.6551556...	0.005545...	-11.254176...
4	0.2,class_1	-0.596609...	-0.473152...	-0.271663...	0.0881968...	-0.008100...	0.0100584...	0.4442748...	0.0461394...	0.3543971...	-0.823593...	0.3189344...	0.042654...	-0.004917...	13.440659...
5	0.2,class_2	0.1815346...	0.2365301...	0.068344...	0.0252516...	0.0047439...	-0.063012...	-0.781539...	0.0325302...	-0.220801...	0.4927216...	-0.261593...	-0.382415...	-0.001011...	-3.285747...
6	0.3,class_0	0.508803...	0.4742831...	0.4421202...	-0.204471...	-0.009465...	0.1212106...	0.595993...	0.0326510...	-0.105881...	-0.071582...	-0.009596...	0.7751128...	0.005858...	-12.25039...
7	0.3,class_1	-0.676053...	-0.440069...	-0.443323...	0.1090031...	-0.012582...	0.080382...	0.485050...	0.0630709...	0.429908...	-0.882908...	0.3500577...	-0.073690...	-0.004990...	15.057767...
8	0.3,class_2	0.3528183...	0.1916765...	0.062295...	0.002454...	0.009666...	-0.148211...	-0.880518...	-0.019634...	-0.227933...	0.54808045	-0.372927...	-0.462612...	-0.001391...	-4.722913...
9	0.4,class_0	0.5176720...	0.495092...	0.5099681...	-0.212669...	-0.008225...	0.1614371...	0.633956...	0.0547798...	-0.173250...	-0.092147...	-0.037370...	0.7312588...	0.005986...	-12.47944...
10	0.4,class_1	-0.751331...	-0.486713...	-0.615199...	0.1461240...	-0.017895...	0.1073317...	0.473380...	0.0856587...	0.4401364...	-1.037478...	0.4194984...	-0.205756...	-0.004856...	17.077453...
11	0.4,class_2	0.3905410...	0.185739611	-0.019380...	-0.003966...	0.0142591...	-0.239384...	-0.910558...	-0.061495...	-0.234150...	0.603473...	-0.419494...	-0.558200...	-0.001529...	-5.012321...
12	0.5,class_0	0.5396071...	0.5010513...	0.588408...	-0.222340...	-0.005666...	0.2257675...	0.643325...	0.0737113...	-0.251388...	-0.102161...	-0.070493...	0.7178426...	0.005863...	-12.88200...
13	0.5,class_1	-0.822937...	-0.475245...	-0.780851...	0.1616994...	-0.023218...	0.1094320...	0.5104253...	0.094080...	0.4927612...	-1.1459118...	0.485996...	-0.265425...	-0.005159...	19.248535...
14	0.5,class_2	0.4197376...	0.1704883...	-0.038732...	-0.005700...	0.02077702	-0.266413...	-1.027899...	-0.090775...	-0.345033...	0.6504755...	-0.456877...	-0.636133...	-0.001128...	-5.934356...
15	0.6,class_0	0.5631989...	0.5050901...	0.6681922...	-0.232196...	-0.002941...	0.2931626...	0.6490755...	0.092238...	-0.330894...	-0.110812...	-0.104292...	0.7082733...	0.0057088...	-13.30635...
16	0.6,class_1	-0.789810...	-0.495626...	-0.965038...	0.1769983...	-0.027208...	0.0769923...	0.509356...	0.1002545...	0.5706862...	-1.2123881	0.529264...	-0.290806...	-0.005308...	19.779707...
17	0.6,class_2	0.4103931...	0.1607048...	-0.013139...	-0.000907...	0.024055...	-0.315353...	-1.066853...	-0.123581...	-0.404860...	0.6997123...	-0.460269...	-0.701821...	-0.001008...	-6.259588...
18	0.7,class_0	0.584620...	0.5086115...	0.7446209...	-0.241625...	-0.000277...	0.3561718...	0.6551811...	0.1104339...	-0.409053...	-0.118854...	-0.1373411...	0.699888...	0.005554...	-13.69481...
19	0.7,class_1	-0.802259...	-0.480391...	-1.126644...	0.1802639...	-0.027764...	0.069295...	0.509968...	0.1311424...	0.5792840...	-1.209617...	0.5742611...	-0.248820...	-0.005465...	20.25034...
20	0.7,class_2	0.389582...	0.2088313...	0.0201889...	-0.022210...	0.033009...	-0.331038...	-1.200698...	-0.144843...	-0.431511...	0.8134227...	-0.504217...	-0.756881...	-0.001112...	-6.804325...
21	0.8,class_0	0.5741750...	0.499800...	0.7616272...	-0.242704...	0.0016796...	0.3274174...	0.6762303...	0.1231733...	-0.459919...	-0.108909...	-0.15181307	0.7197606...	0.0053519...	-13.57458...
22	0.8,class_1	-0.819244...	-0.464545...	-1.285904...	0.1830662...	-0.028195...	0.058466...	0.509603...	0.1639480...	0.584990...	-1.203943...	0.6189363...	-0.199646...	-0.005618...	20.749763...
23	0.8,class_2	0.369284...	0.237863...	0.0492447...	-0.043059...	0.0374500...	-0.276802...	-1.319856...	-0.150054...	-0.473120...	0.852594...	-0.505502...	-0.743021...	-0.000963...	-6.980339...
24	0.9,class_0	0.5861951...	0.485245...	0.8122537...	-0.245589...	0.0041198...	0.293565...	0.7030133...	0.1387402...	-0.513100...	-0.097008...	-0.155748...	0.7433095...	0.005055...	-13.77380...
25	0.9,class_1	-0.832150...	-0.449737...	-1.4471271...	0.1866963...	-0.029073...	0.0507253...	0.5157923...	0.1933818...	0.60073471	-1.205362...	0.6647788...	-0.158809...	-0.005811...	21.283117...
26	0.9,class_2	0.3491110...	0.266885...	0.0780853...	-0.063917...	0.0418858...	-0.222320...	-1.439123...	-0.155259...	-0.514499...	0.8915821...	-0.506783...	-0.729244...	-0.000812...	-7.1571245...
27	1,class_0	0.6193942...	0.4681769...	0.9125898...	-0.252243...	0.0066105...	0.2728150...	0.7249734...	0.1575158...	-0.574424...	-0.093633...	-0.1517104...	0.7620898...	0.0047426...	-14.23054...
28	1,class_1	-0.857660...	-0.450495...	-1.605953...	0.2016596...	-0.031994...	0.085035...	0.528694...	0.21541827	0.6815130...	-1.258292...	0.7185657...	-0.189570...	-0.006334...	22.395796...
29	1,class_2	0.320994...	0.2870073...	0.0591227...	-0.075430...	0.0447507...	-0.197533...	-1.500229...	-0.149029...	-0.543082...	0.9203241...	-0.509806...	-0.712545...	-0.000774...	-7.015709...

Figure 6: Coefficient and intercept of SVM with slack variables

Indices of Support Vectors See Figure 7 for the indices. And we can know that,

	index	class_0	class_1	class_2
0	0.1	6,15,42,78,96,112,130,136,3,8,44,84,87,99,124	2,3,8,12,44,55,73,102,104,122,124,125,132,17,19,40,53,75,78,96,112,119,121,129,130	9,17,23,32,40,93,100,121,129,10,15,73,104,125,132,137
1	0.2	6,75,78,96,112,130,136,3,8,44,84,87,124	2,8,12,44,55,73,104,122,124,132,17,19,40,53,78,112,119,121,130	17,32,40,62,93,100,121,10,73,104,125,132
2	0.300000...	6,78,96,112,114,130,136,3,8,44,84,87,124	2,8,44,55,73,104,122,124,132,17,19,40,53,78,112,119,121,130	17,32,40,50,62,93,100,121,10,48,73,125,132
3	0.4	6,78,96,112,114,130,136,3,8,44,84,87,124	2,8,44,55,73,104,122,132,17,19,40,42,53,112,119,121	17,40,50,62,93,100,121,10,48,73,104,125,132
4	0.5	6,78,96,112,114,130,136,3,8,44,87,124	2,8,44,48,55,73,122,132,17,40,53,78,112,119,121	17,40,93,100,121,10,48,73,104,125,132
5	0.600000...	6,78,96,112,114,130,136,3,8,44,87,124	2,8,44,48,55,73,122,132,17,40,53,78,112,119,121	40,93,100,121,10,48,73,79,104,125,132
6	0.700000...	6,78,96,112,114,130,136,3,8,44,87,124	2,8,44,48,55,73,122,132,17,19,40,53,78,112,119,121	40,93,100,121,10,48,73,79,104,125,132
7	0.8	6,78,112,114,130,136,3,8,44,87,124	2,8,44,48,55,73,122,132,17,19,40,53,78,112,119,121	40,93,100,121,10,48,73,79,104,125,132
8	0.9	6,78,112,114,130,136,3,44,124	2,8,44,48,55,73,122,132,17,19,40,53,78,112,121	40,93,100,121,10,48,73,79,104,125,132
9	1.0	6,78,112,114,130,136,3,44,124	2,8,44,48,73,122,132,17,19,40,53,78,112,121	40,83,93,100,121,10,48,73,79,104,125,132

Figure 7: Support vector indices in SVM with slack variables

increasing the strength of the penalty will somehow decrease the number of support vectors. This is because, when we increase the penalty, the decision boundary becomes "closer", and hence fewer points fall on the edge. Therefore, it decreases the number of support vectors.

Slack Variables First we show the overall slack variables. See Figure 8[2] In order

	index	2	3	6	8	10	12	15	17	19	32	40	42	44
0	0.1,class_0	nan	nan	0.002384...	0.5985778...	nan	nan	nan	nan	nan	nan	nan	0.002393...	1.9684954...
1	0.1,class_1	0.1894534...	0.0001176...	nan	0.9196408...	nan	0.0081829...	nan	0.939644...	0.2795758...	nan	1.53771511...	nan	1.2749104...
2	0.1,class_2	nan	nan	nan	nan	0.4146589...	nan	0.098482...	0.0026576...	nan	0.1376103...	1.3736275...	nan	nan
3	0.2,class_0	nan	nan	0.004363...	0.088795...	nan	nan	nan	nan	nan	nan	nan	nan	1.6661228...
4	0.2,class_1	0.1192288...	nan	nan	1.0594413...	nan	0.0148933...	nan	0.7137033...	0.003465...	nan	1.3126137...	nan	1.3969150...
5	0.2,class_2	nan	nan	nan	nan	0.3567116...	nan	nan	nan	nan	nan	1.2975261...	nan	nan
6	0.3,class_0	nan	nan	0.008469...	0.0228621...	nan	nan	nan	nan	nan	nan	nan	nan	1.6960360...
7	0.3,class_1	0.0264793...	nan	nan	0.9358612...	nan	nan	nan	0.629062...	0.0059316...	nan	1.1322946...	nan	1.2935621...
8	0.3,class_2	nan	nan	nan	nan	0.1986741...	nan	nan	nan	nan	nan	1.2880925...	nan	nan
9	0.4,class_0	nan	nan	0.0109408...	0.0298312...	nan	nan	nan	nan	nan	nan	nan	nan	1.6497371...
10	0.4,class_1	0.0097230...	nan	nan	0.8421549...	nan	nan	nan	0.2706674...	0.0061983...	nan	1.1139270...	0.0151845...	1.2346483...
11	0.4,class_2	nan	nan	nan	nan	0.1161489...	nan	nan	nan	nan	nan	1.25395388	nan	nan
12	0.5,class_0	nan	nan	0.0129655...	0.0332731...	nan	nan	nan	nan	nan	nan	nan	nan	1.5908851...
13	0.5,class_1	0.0109585...	nan	nan	0.8361474...	nan	nan	nan	0.080266...	nan	nan	0.9071059...	nan	1.1451935...
14	0.5,class_2	nan	nan	nan	nan	0.0156544...	nan	nan	nan	nan	nan	0.939766...	nan	nan
15	0.6,class_0	nan	nan	0.0150189...	0.036309...	nan	nan	nan	nan	nan	nan	nan	nan	1.5304909...
16	0.6,class_1	0.0114576...	nan	nan	0.8574567...	nan	nan	nan	nan	nan	nan	0.7803766...	nan	1.0316624...
17	0.6,class_2	nan	nan	nan	nan	0.0114589...	nan	nan	nan	nan	nan	0.804209...	nan	nan
18	0.7,class_0	nan	nan	0.0170984...	0.039252...	nan	nan	nan	nan	nan	nan	nan	nan	1.4716033...
19	0.7,class_1	0.0110689...	nan	nan	0.8710406...	nan	nan	nan	nan	0.0021527...	nan	0.6861935...	nan	0.9410059...
20	0.7,class_2	nan	nan	nan	nan	0.0117907...	nan	nan	nan	nan	nan	0.582695...	nan	nan
21	0.8,class_0	nan	nan	0.0191104...	0.0406981...	nan	nan	nan	nan	nan	nan	nan	nan	1.4317051...
22	0.8,class_1	0.0106692...	nan	nan	0.8877081...	nan	nan	nan	nan	0.000860...	nan	0.59473088	nan	0.8531040...
23	0.8,class_2	nan	nan	nan	nan	0.0110874...	nan	nan	nan	nan	nan	0.3507399...	nan	nan
24	0.9,class_0	nan	nan	0.0209723...	0.0180933...	nan	nan	nan	nan	nan	nan	nan	nan	1.3550800...
25	0.9,class_1	0.0109622...	nan	nan	0.91117810...	nan	nan	nan	nan	nan	nan	0.484834...	nan	0.768454...
26	0.9,class_2	nan	nan	nan	nan	0.0100441...	nan	nan	nan	nan	nan	0.1186355...	nan	nan
27	1,class_0	nan	nan	0.022608...	nan	nan	nan	nan	nan	nan	nan	nan	nan	1.2355246...
28	1,class_1	0.0104671...	nan	nan	0.8579355...	nan	nan	nan	nan	nan	nan	0.3065103...	nan	0.6764614...
29	1,class_2	nan	nan	nan	nan	0.0150680...	nan	nan	nan	nan	nan	0.0012050...	nan	nan

Figure 8: (Part of) slack variables

to understand the changes more directly, we visualize the results. See Figure 9 and 10. We observe that, as the strength of the penalty increases, the number and sum of slack variables will decrease. The reasons are quite the same as previously. The narrow "decision boundary" attributes to the increase in the penalty. And since the boundary becomes more narrow, the model tends to use fewer slack variables, and the values tend to be lower.

2.4 SVM with Non-linear Kernels

2.4.1 Optimization Problem

Since the data might not be linearly separable, we use non-linear kernel functions to map the original data to $f(\cdot)$, where f is the kernel function.

The mapping procedure is

$$x \to \phi(x)$$

[2] Since the slack variable matrix is (30, 40). Here I cannot show all the details of the slack variable values. Please refer to the jupyter notebook for details. I report the number and the sum of slack variables in the notebook.

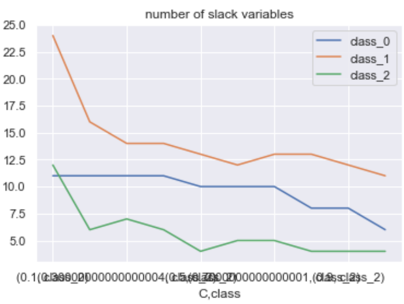

Figure 9: Number of slack variables

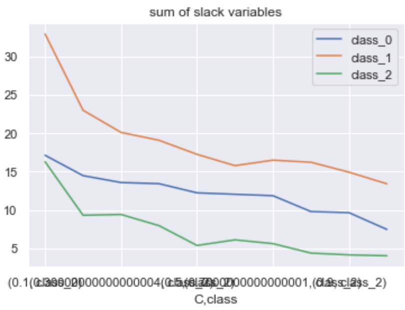

Figure 10: Sum of slack variable values

Then we define the kernel as

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi\left(\mathbf{x}_i\right)^{\top} \phi\left(\mathbf{x}_j\right)$$

Therefore, we have an optimization problem (quite similar to Equation 3),

$$\begin{aligned} &\min_{\mathbf{w}, b, \xi} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i^m \xi_i \\ &\text{s.t. } 1 - \xi_i - y_i\left(\mathbf{w}^{\top} \phi(\mathbf{x}_i) + b\right) \leq 0, \quad -\xi_i \leq 0, \quad \forall i \end{aligned} \tag{4}$$

Next, we derive the Lagrangian function of Equation 4,

$$\mathscr{L}(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C \sum_i^m \xi_i + \sum_i^m \alpha_i \left(1 - \xi_i - y_i\left(w^T \phi\left(x_i\right) + b\right) + \beta_i\left(-\xi_i\right)\right) \tag{5}$$

where $\alpha_i \geq 0$ and $\beta_i \geq 0$

Based on the Lagrangian function, we calculate the partials to meet the K.K.T. conditions,

$$\begin{aligned} \frac{\partial L}{\partial w} &= 0 \Longrightarrow w = \sum_i^m \alpha_i y_i \phi\left(x_i\right) \\ \frac{\partial L}{\partial b} &= 0 \Longrightarrow 0 = \sum_i^m \alpha_i y_i \\ \frac{\partial L}{\partial \xi_i} &= 0 \Longrightarrow \alpha_i = C - \beta_i, \forall i \end{aligned}$$

Finally, we have our dual problem,

$$\begin{aligned} &\max_{\alpha} \sum_i^m \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k\left(\mathbf{x}_i, \mathbf{x}_j\right) \\ &\text{s.t. } \sum_i^m \alpha_i y_i = 0, \alpha_i \geq 0, \forall i \end{aligned} \tag{6}$$

In this question, we use three different kernels,

- Polynomial kernel: $k\left(\mathbf{x}, \mathbf{x}_i\right) = \left(1 + \frac{\mathbf{x}^{\top}\mathbf{x}_i}{\sigma^2}\right)^p, p > 0$

- Radial basis function kernel: $k(\mathbf{x}, \mathbf{x}_i) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}\right\}$
- Sigmoid kernel: $k(\mathbf{x}, \mathbf{x}_i) = \frac{1}{1+\exp^{-\frac{x^\top x_j+0}{\sigma^2}}}$

Now we have already shown the sketch of the optimization problem, then we just show the results, and analyze the effect of different kernels.

2.4.2 Results and Analysis

Bias Terms (Intercepts) See Table 2 for details.

Table 2: Bias terms in SVM with non-linear kernels

Models	**class_0**	**class_1**	class_2
2-order poly	-2.78092492	1.68456508	-1.00251071
3-order poly	-2.28252152	1.29640558	-1.00163291
RBF model	-0.06468105	0.06169	-1.13598704
Sigmoid model	-14.15164655	13.67357302	6.17347792

Errors of SVM with Non-linear Kernels See Table 3 for details.

Table 3: Errors of SVM with non-linear kernels

Models	**training_error**	**testing_error**
2-order poly	0.302816901408450740	0.33333333333333337
3-order poly	0.316901408450704250	0.33333333333333337
RBF model	0.302816901408450740	0.33333333333333337
Sigmoid model	0.7816901408450705	0.7777777777777778

Indices of Support Vectors See below Figure (11-14) for details.

```
Indices of support vectors in Class 0: [ 10  15  17  24  41  42  48  65  72  78  95  96 112 130 136   0   3   8
  12  14  18  27  43  44  69  77  84  87  99 124]
Indices of support vectors of Class 1: [  2   4  10  13  15  16  18  22  24  25  28  29  30  31  33  36  45  48
  51  61  72  73  79  84  87  91  95  99 102 104 105 107 110 122 123 124
 125 126 132 136 137   6   9  17  19  23  26  32  39  40  41  42  46  47
  50  53  60  62  63  67  75  78  81  85  93  96  98 103 106 109 111 112
 113 117 119 121 127 128 129 130 134 140]
Indices of support vectors of Class 2: [  5   9  19  21  23  26  32  40  44  46  47  60  62  63  66  67  69  76
  78  80  81  82  84  85  87  90  93  94  99 100 101 103 111 112 114 115
 116 118 124 128 129 141   2   4  10  13  15  16  22  24  25  28  29  30
  31  33  36  45  48  51  61  65  72  73  79  91  95 102 104 105 107 110
 122 123 125 126 132 136 137]
```

Figure 11: 2-order polynomial kernel

```
Indices of support vectors in Class 0: [ 10  15  17  24  41  42  48  65  78  95  96 112 136   3   8  14  18  27
  43  44  69  77  84  87  99 124]
Indices of support vectors of Class 1: [  2   4  10  13  15  16  18  22  24  25  28  29  30  31  33  36  45  48
  51  61  72  73  79  84  87  91  95  99 102 104 105 107 110 122 123 124
 125 126 132 136 137   6   9  17  19  23  26  32  39  40  41  42  46  47
  50  53  60  62  63  67  75  78  81  85  93  96  98 103 106 109 111 112
 113 117 119 121 127 128 129 130 134 140]
Indices of support vectors of Class 2: [  5   9  14  17  18  19  23  26  32  40  44  46  47  53  60  62  63  66
  67  69  76  81  84  85  87  90  93  94  99 103 109 111 112 114 115 116
 118 119 124 128 129 134   2   4  10  13  15  16  22  24  25  28  29  30
  31  33  36  45  48  51  61  65  72  73  79  91  95 102 104 105 107 110
 122 123 125 126 132 136 137]
```

Figure 12: 3-order polynomial kernel

```
Indices of support vectors in Class 0: [ 10  15  17  24  28  36  41  42  48  65  72  78  95  96 112 121 130 136
   0   3   8  12  14  18  27  43  44  55  68  69  77  84  87  99 124 133]
Indices of support vectors of Class 1: [  2   4  13  15  16  18  22  24  25  28  29  30  31  33  35  36  45  48
  51  61  72  73  79  84  87  91  95  99 102 104 105 107 110 122 123 124
 125 126 132 136 137   6   9  17  19  23  26  32  39  40  41  42  46  47
  50  53  60  62  63  67  75  78  81  85  93  96  98 103 106 109 111 112
 113 117 119 121 127 128 129 130 134 140]
Indices of support vectors of Class 2: [  6   8   9  18  19  26  32  39  40  41  42  44  46  47  50  53  63  67
  75  77  81  84  85  87  93  99 103 106 109 111 113 117 119 121 124 129
 130 134   2   4  10  13  15  16  22  24  25  28  29  30  31  33  36  45
  48  51  61  65  72  73  79  91  95 102 104 105 107 110 122 123 125 126
 132 136 137]
```

Figure 13: RBF kernel

```
Indices of support vectors in Class 0: [  2   4   6   9  13  16  19  22  25  26  29  31  32  33  39  40  45  46
  47  50  51  53  61  63  67  73  75  79  81  85  91  93 103 104 105 106
 107 109 110 111 113 117 122 125 126 129 132 137   0   1   3   7   8  11
  12  14  18  27  34  35  37  38  43  44  49  52  54  55  57  58  64  66
  68  69  70  71  74  77  83  84  87  88  89  97  99 100 101 108 120 124
 131 133 135 138 139 141]
Indices of support vectors of Class 1: [  0   1   3   7   8  10  11  12  14  15  18  24  27  34  35  37  38  43
  44  49  52  54  55  57  58  64  65  66  68  69  70  71  74  77  83  84
  87  88  89  97  99 100 101 102 108 120 131 133 135 138 139 141   5   6
   9  17  19  21  23  26  32  39  40  41  42  46  47  50  53  56  59  60
  62  63  67  75  76  78  80  81  85  90  93  94  96  98 103 106 109 111
 113 114 115 116 117 118 119 121 127 128 129 130 134 140]
Indices of support vectors of Class 2: [  0   1   7  11  12  27  34  35  37  38  49  52  54  55  57  58  64  66
  68  70  71  74  83  88  89  97 100 101 108 112 120 131 133 135 138 139
 141   2   4  10  13  15  16  22  24  25  28  29  30  31  33  36  45  48
  51  61  65  72  73  79  91  95 102 104 105 107 110 122 123 125 126 132
 136 137]
```

Figure 14: Sigmoid kernel

Summary From the above results, we can have the following observations: 1) the **errors of the linear kernel SVM** is minimum among all these kernels, which means this dataset is linearly separable. And for the non-linear kernels, the testing errors are somehow large. 2) Since we have 13 attributes (features) and 178 input data (in lines), the number of features seems to be comparable to the input data. Hence, the dataset is more likely to be linearly separable. 3) 2-, 3-order polynomial kernel, and the RBF kernel SVMs produce similar errors (including both testing and training errors). This might because the kernels maps the data in a very similar way, and hence we train similar SVM models.